IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 7, 2018, accepted December 26, 2018, date of publication January 23, 2019, date of current version February 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894289

COMMENTS AND CORRECTIONS

Correction to "Improving Privacy and Security in Decentralizing Multi-Authority Attribute-Based Encryption in Cloud Computing"

SYH-YUAN TAN
School of Computing, Newcastle University, Newcastle upon Tyne, NE4 5TG, U.K.
e-mail: syh-yuan.tan@newcastle.ac.uk

This work was supported in part by the European Research Council Starting Grant "Confidentiality-Preserving Security Assurance (CASCAde)," under Grant GA n°716980.

ABSTRACT In 2018, Yang *et al*. proposed a decentralized multi-authority attribute-based encryption scheme for cloud computing applications and proved its security using the dual system encryption technique. In this comment, we show that Yang *et al*.'s scheme does not achieve encryption one-wayness under the key-only attack and the user collusion attack, respectively. In the key-only attack, with the knowledge of public parameters only, an adversary can impersonate the attribute authorities to forge user attribute secret keys. In the user collusion attack, malicious users can collude by sharing their secret keys to unauthorizedly decrypt a ciphertext. In order to fix the scheme, we suggest adopting a pairing-based proof of knowledge protocol and the decryption algorithm from Lewko and Water's ABE scheme.

INDEX TERMS Cryptanalysis, decentralizing, multi-authority, attribute-based encryption, dual system encryption.

I. INTRODUCTION

In 2018, Yang *et al*. [4] designed a decentralizing multi-authority attribute-based encryption (D-MA-ABE) scheme for cloud computing applications. The D-MA-ABE scheme solves the single point of failure problem in the traditional ABE scheme and at the same time provides anonymity property for users. The latter is achieved by setting an attribute authority AA_i to request the attribute keys from other attribute authorities AA_j on behalf of its users, if AA_i does not has the attributes needed. In such setting, when the public keys of every *AA* are made public, the D-MA-ABE scheme allows dynamic joining of *AA*s.

Although the security of D-MA-ABE scheme is assured by Yang *et al*. using the dual system encryption technique, we discover two security flaws in their scheme. The first flaw occurs in the key issuing protocol which is the core engine for the dynamic joining of *AA*. The protocol is represented by the second and third processes in Figure 1 which include the verification of requester identity and the generated secret key. Specifically, we discover that using the knowledge of public parameters only, an adversary can impersonate any AA_i to request the user attribute secret keys from another AA_j. In the extreme case, the flaw allows the adversary to collect all attribute keys from the *AA*s and own the D-MA-ABE system such that the adversary can decrypt any given ciphertext. This shows that Yang *et al*.'s D-MA-ABE scheme does not achieve encryption one-wayness under the key-only attack. The second flaw resides in the ciphertext construction where by the access control matrix is not properly bound to the secret exponents. This allows malicious users to unauthorizedly decrypt a ciphertext as their unique secret key component which should stop them from colluding, can be cancelled out during the decryption process. In order to fix the scheme, we suggest to adopt a proof of knowledge protocol as well as the decryption algorithm from Lewko and Water's ABE [2] scheme.

This comment paper is organized as follow. We briefly describe the mathematical background related to Yang *et al*.'s D-MA-ABE scheme in Section II and present the attacks in Section III. Finally, we suggest some solutions to the attacks in Section IV.

II. PRELIMINARIES

A. BILINEAR MAP

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two distinct cyclic groups of prime order q and let g_1, g_2 be the point generators in $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is said to be an admissible map if it fulfills the following conditions:

1) Bilinearity:
   - The placements of the scalar multipliers of the two points to e does not affect the mapping result.
   - Example: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$ for all $a, b \in \mathbb{Z}_q$.
2) Non-degeneracy:
   - The result of the mapping is unique.
   - Example: $e(g_1, g_1) \neq e(g_2, g_2) \neq e(g_1, g_2) \neq 1$.
3) Efficiently Computable: e can be easily computed given any two points g_1 and g_2.

Yang *et al.*'s D-MA-ABE uses a variant of bilinear map, namely, composite order bilinear map whose order $q = N = p_1p_2p_3$ is a composite number and is constructed by three primes p_1, p_2, p_3. We can, however, view their composite order bilinear map the same as a prime order bilinear map as their D-MA-ABE operates in the subgroup $\mathbb{G}_{p_1}$ only [4].

B. LINEAR SECRET SHARING SCHEME

A linear secret sharing scheme (LSSS) fulfills the following conditions:

1) The shares for each party generated a vector $\lambda \in \mathbb{Z}_q^{|\lambda|}$.
2) There exists a matrix A of n rows and l columns which can generate the shares. Each row $i \in \{1, \ldots, n\}$ is labeled by a function ρ such that $\rho(i)$ denotes the i-th participant. Given a vector $v = (s, v_2, \ldots, v_l) \in \mathbb{Z}_q^l$ where s is the secret to be shared and $v_2, \ldots, v_l$ are chosen randomly, $\lambda = A \cdot v = (\lambda_1, \lambda_2, \ldots, \lambda_l)$ is the vector of shares for the secret s. We can also say that $\lambda_i = A_i \cdot v$ is the share of the participant $\rho(i)$.

Given a LSSS represented by (A, ρ) as above, there exists constants $C = (c_1, \ldots, c_{|I|}) \in \mathbb{Z}_q^{|I|}$ such that $s = \sum_{i \in I} c_i \lambda_i$ can be reconstructed if and only if $\mathbb{A}(S) = 1$ where I is the set of rows $\{i | \rho(i) \in S\}$.

C. ACCESS TREE

When the threshold in LSSS is set to t out of total n, it becomes a (t, n)-gate in which the secret can be recovered when at least t shares are presented. We can see that if $t = n$, the LSSS becomes an AND-gate; if $t = 1$, the LSSS becomes an OR-gate. Combining these gates recursively, we can build an access tree $\mathbb{A}$. Given an attribute set S, $\mathbb{A}(S) = 1$ if it is satisfied by S while $\mathbb{A}(S) = 0$ when it is not satisfied by S.

In order to illustrate the use of LSSS in constructing an $\mathbb{A}$, we modify the converting algorithm from [3] to efficiently convert an access tree to a compressed Shamir's LSSS [1] matrix. Let u be the vector which contains the node label and k be the counter of the branches, the converting algorithm goes down the tree in depth-first search manner, and labels the leaf nodes as follows:

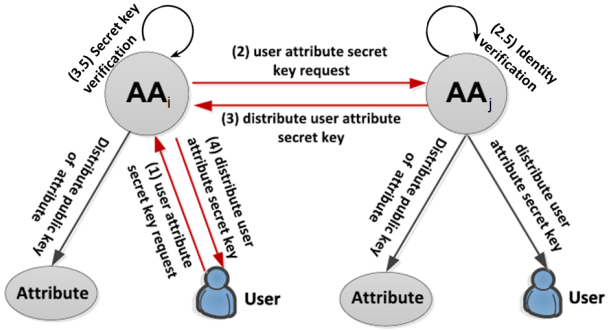

FIGURE 1. Key Distribution Process of Yang *et al.*'s D-MA-ABE Scheme [4].

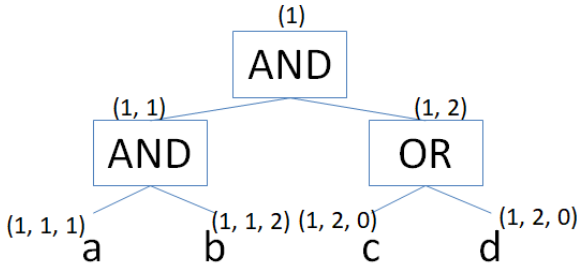

FIGURE 2. Converting an Access Tree to a LSSS Matrix.

1) if the current node is the root node, set $u = (1), k = 0$ and traverse to the left branch if there is any.
2) if the parent node is an AND gate, increment k by 1 and concatenate it to the label followed by the index $1 \leq i \leq n$ of attributes, e.g., $u = (1, k, i)$ where n is the total attributes under the branch. If $n > 1$, increment i and record each index in a new row.
3) if the parent node is an OR gate, increment k by 1 and concatenate it to the label followed by the number 0, i.e., regardless of the index $1 \leq i \leq n$ such that $u = (1, k, 0)$. If $n > 1$, duplicate the row for n times.
4) when all nodes are labeled, pad the shorter rows with number 0 so that each row has the same length.

Considering the access tree $\mathbb{A} = \{\{a \text{ AND } b\} \text{ AND } \{c \text{ OR } d\}\}$ shown in Figure 2. The generated 6×3 full LSSS matrix A can be simplified into a 4×3 matrix to provide only the essential information needed by the user:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \end{bmatrix} \begin{array}{l} \rightarrow \text{left AND gate} \\ \quad \rightarrow \rho(1) = a \\ \quad \rightarrow \rho(2) = b \\ \rightarrow \text{right OR gate} \\ \quad \rightarrow \rho(3) = c \\ \quad \rightarrow \rho(4) = d \end{array} \Longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \end{bmatrix}$$

We can compress the matrix by substituting the indexes in the access tree with their corresponding attributes as below:

$$A = \begin{bmatrix} 1 & 1 & a \\ 1 & 1 & b \\ 1 & c \cdot d & 0 \end{bmatrix}$$

This reduces the rows in A whenever there is an OR gate and also simplifies the mapping function ρ which is now implicitly encoded into A as each row can be uniquely identified using the attribute in it. Since the matrix A has $l = 3$ columns, the secret vector $v \in \mathbb{Z}_q^l$ has 3 elements with the secret s to be shared placed at the first. The constant c_i can be calculated from the equation:

$$c_i = \prod_{j \in S, j \neq i} \frac{0-j}{i-j}$$

where S is the attribute set under the current branch while i and j are the indexes (or attributes if compressed) in the set S.

Yang *et al.*'s D-MA-ABE abuses the notation in the process of recovering s such that the recursive process is ignored [4], as if the access tree is only a single level access tree which is exactly a LSSS. We note that this is a common practice for the ABE schemes described using LSSS matrices in order to simplify the scheme description.

III. CRYPTANALYSIS

In this section, we present the key-only attack and the user collusion attack on Yang *et al.*'s D-MA-ABE scheme to break its encryption one-wayness property, respectively. Since this is a comment paper, we do not recall the details of the D-MA-ABE [4] scheme.

A. KEY-ONLY ATTACK

In order to ease our explanation, we consider the following scenario. A malicious user A wants to decrypt the victim user V's ciphertext C. User A's attribute secret key contains the attributes $\{a, b\} = \{IT, admin\}$ which are generated by the attribute authority AA_1; while user V's attribute secret key contains the attributes $\{c, d\} = \{finance, manager\}$ which are generated by the attribute authority AA_2. Assuming AA_1 and AA_2 handle the attributes $\{a, b\}$ and $\{c, d\}$ respectively, and the access structure on C is set to $\{c$ AND $d\} = \{finance$ AND $manager\}$, we explain how user A decrypts C successfully as follow.

1) We first display the public parameters known to user A:

$$\begin{aligned}
&ID_{AA_1}, ID_a = ID_{AA_1}||IT, ID_b = ID_{AA_1}||admin,\\
&ID_{AA_2}, ID_c = ID_{AA_2}||finance, ID_d = ID_{AA_2}||manager,\\
&PK_{AA_1} = g^{s_1 H_1(ID_{AA_1})},\\
&PK_a = \{PK'_a = e(g_1, g_1)^{s_1(H_{x_1}(ID_a))},\\
&\qquad PK''_a = g_1^{H_{x_1}(ID_a)}\},\\
&PK_b = \{PK'_b = e(g_1, g_1)^{s_1(H_{x_1}(ID_b))},\\
&\qquad PK''_b = g_1^{H_{x_1}(ID_b)}\},\\
&PK_{AA_2} = g^{s_2 H_1(ID_{AA_2})},\\
&PK_c = \{PK'_c = e(g_1, g_1)^{s_2(H_{x_2}(ID_c))},\\
&\qquad PK''_c = g_1^{H_{x_2}(ID_c)}\},\\
&PK_d = \{PK'_d = e(g_1, g_1)^{s_2(H_{x_2}(ID_d))},\\
&\qquad PK''_d = g_1^{H_{x_2}(ID_d)}\}
\end{aligned}$$

2) Next, from PK_{AA_1}, user A can compute the value:

$$X_1 = \left(g^{s_1 H_1(ID_{AA_1})}\right)^{1/H_1(ID_{AA_1})} = g^{s_1}$$

3) Using X_1, User A can impersonate AA_1 to request the attribute secret key $SK_{c,A}$ for the attribute $c = finance$ from AA_2 as follows:
   a) User A sends $H_2(GID) = H_2(ID_{AA_1}||ID_A)$, $ID_c = ID_{AA_2}||finance, ID_{AA_1}$ to AA_2, together with $C_A = X_1^{H_1(ID_c)} H_2(GID)$. Note that the prefix in ID_c is not ID_{AA_1} because the attribute $c = finance$ is handled by AA_2.
   b) AA_2 accepts the request since[1]:

$$\begin{aligned}
&e(g_1, C_A)^{H_1(ID_{AA_1})}\\
&= e(g_1, X_1^{H_1(ID_c)} H_2(GID))^{H_1(ID_{AA_1})}\\
&= e(g_1, g^{s_1 H_1(ID_c)} H_2(GID))^{H_1(ID_{AA_1})}\\
&= e(g_1, PK_1)^{H_1(ID_c)} e(g_1, H_2(GID))^{H_1(ID_{AA_1})}
\end{aligned}$$

   c) AA_2 computes and sends

$$SK_{c,A} = g_1^{s_2 H_{x_2}(ID_c)} H_2(GID)^{H_{x_2}(ID_c)} H_2(ID_{AA_1})$$

   to user A who is mistaken as AA_1.

4) User A repeats the same request for $SK_{d,A}$ using X_1.
5) Given the ciphertext C as:

$$\begin{aligned}
&C_0 = Me(g_1, g_1)^s, \quad C_{1,x} = e(g_1, g_1)^{\lambda_x} (PK'_x)^{r_x},\\
&C_{2,x} = g_1^{r_x}, \quad C_{3,x} = g_1^{r_x} g_1^{\omega_x}, \quad C_{4,x} = (PK''_x)^{r_x} g_1^{\omega_x}
\end{aligned}$$

where $\omega = \{0, \omega'\}, v = \{s, v'\} \in \mathbb{Z}_N^2$, $\omega_x = A_x \cdot \omega$ and $\lambda_x = A_x \cdot v$ with A as the 2×2 access matrix for $x \in \{c, d\}$. User A can decrypt C using $SK_{x,A}$ by calculating $\frac{C_{1,x} e(H_2(GID), C_{4,x}) e(H_2(ID_1), C_{3,x})}{e(SK_{x,A}, C_{2,x})}$, as shown at the top of the next page, to find M such that:

$$\begin{aligned}
&\frac{C_0}{\prod_x \left(e(g_1, g_1)^{\lambda_x} e(H_2(GID), g_1)^{\omega_x} e(H_2(ID_{AA_1}), g_1)^{\omega_x}\right)^{c_x}}\\
&= \frac{Me(g_1, g_1)^s}{e(g_1, g_1)^s}\\
&= M
\end{aligned}$$

where $\sum_x c_x A_x v = s$ and $\sum_x c_x A_x \omega = 0$ for the constants $c_x \in \mathbb{Z}_N$.

B. COLLUSION ATTACK

In order to illustrate the collusion attack, we consider the scenario of two malicious users A and B collude to unauthorizedly decrypt a ciphertext C. User A's attribute secret key contains the attributes $\{a, b\} = \{IT, admin\}$ which are generated by the attribute authority AA_1; while user B's attribute secret key contains the attributes $\{c, d\} = \{finance, manager\}$ which are generated by the attribute authority AA_2. Assuming AA_1 and AA_2 handle the attributes

[1] We believe the equation in step 3 and step 5 of the key issuing protocol in [4] has typographic errors where the value ID_u at the right hand side should be GID as ID_u is not known to AA_2.

$$
\begin{aligned}
&\frac{C_{1,x}e(H_2(GID), C_{4,x})e(H_2(ID_1), C_{3,x})}{e(SK_{x,A}, C_{2,x})} \\
&= \frac{e(g_1, g_1)^{\lambda_x}(PK'_x)^{r_x}e(H_2(GID), (PK''_x)^{r_x}g_1^{\omega_x})}{e(g_1^{s_2H_{x_2}(ID_x)}H_2(GID)^{H_{x_2}(ID_c)}H_2(ID_{AA_1}), g_1^{r_x})} \\
&\quad \times \frac{e(H_2(ID_{AA_1}), g_1^{r_x}g_1^{\omega_x})}{e(g_1^{s_2H_{x_2}(ID_x)}H_2(GID)^{H_{x_2}(ID_x)}H_2(ID_{AA_1}), g_1^{r_x})} \\
&= \frac{e(g_1, g_1)^{\lambda_x}e(H_2(GID), (PK''_x)^{r_x}g_1^{\omega_x})e(H_2(ID_{AA_1}), g_1^{r_x}g_1^{\omega_x})}{e(H_2(GID)^{H_{x_2}(ID_x)}H_2(ID_{AA_1}), g_1^{r_x})} \\
&= \frac{e(g_1, g_1)^{\lambda_x}e(H_2(GID), g_1^{\omega_x})e(H_2(ID_{AA_1}), g_1^{r_x}g_1^{\omega_x})}{e(H_2(ID_{AA_1}), g_1^{r_x})} \\
&= e(g_1, g_1)^{\lambda_x}e(H_2(GID), g_1)^{\omega_x}e(H_2(ID_{AA_1}), g_1)^{\omega_x} \\
&\frac{C_{1,b}e(H_2(GID_A), C_{4,b}C_{2,b})}{e(SK'_{b,A}, C_{2,b})e(H(GID_A), C_{3,b})} \\
&= \frac{e(g_1, g_1)^{\lambda_b}(PK'_b)^{r_b}e(H_2(GID_A), (PK''_b)^{r_b}g_1^{\omega_b}g_1^{r_b})}{e(g_1^{s_1H_{x_1}(ID_b)}H_2(GID_A)^{H_{x_1}(ID_b)}, g_1^{r_b})e(H(GID_A), g_1^{r_b}g_1^{\omega_b})} \\
&= \frac{e(g_1, g_1)^{\lambda_b}e(g_1, g_1)^{s_1H_{x_1}(ID_b)r_b}e(H_2(GID_A), g_1^{H_{x_1}(ID_b)r_b}g_1^{\omega_b}g_1^{r_b})}{e(g_1, g_1)^{s_1H_{x_1}(ID_b)r_b}e(H_2(GID_A), g_1^{H_{x_1}(ID_b)r_b}g_1^{r_b}g_1^{\omega_b})} \\
&= e(g_1, g_1)^{\lambda_b} \\
&\frac{C_{1,c}e(H_2(GID_B), C_{4,c}C_{2,c})}{e(SK'_{c,B}, C_{2,c})e(H(GID_B), C_{3,c})} \\
&= \frac{e(g_1, g_1)^{\lambda_b}(PK'_c)^{r_c}e(H_2(GID_B), (PK''_c)^{r_c}g_1^{\omega_c}g_1^{r_c})}{e(g_1^{s_2H_{x_2}(ID_c)}H_2(GID_B)^{H_{x_2}(ID_c)}, g_1^{r_c})e(H(GID_B), g_1^{r_c}g_1^{\omega_c})} \\
&= \frac{e(g_1, g_1)^{\lambda_c}e(g_1, g_1)^{s_2H_{x_2}(ID_c)r_b}e(H_2(GID_B), g_1^{H_{x_2}(ID_c)r_c}g_1^{\omega_c}g_1^{r_c})}{e(g_1, g_1)^{s_2H_{x_2}(ID_c)r_c}e(H_2(GID_B), g_1^{H_{x_2}(ID_c)r_c}g_1^{r_c}g_1^{\omega_c})} \\
&= e(g_1, g_1)^{\lambda_c}
\end{aligned}
$$

$\{a, b\}$ and $\{c, d\}$ respectively, and the access structure on C is set to $\{b \text{ AND } c\} = \{admin \text{ AND } finance\}$, we explain how users A and B decrypt C successfully as follow.

1) Firstly, users A and B modify their attribute secret keys as such:

$$
\begin{aligned}
SK'_{b,A} &= SK_{b,A}/H_2(ID_{AA_1}) \\
&= g_1^{s_1H_{x_1}(ID_b)}H_2(GID_A)^{H_{x_1}(ID_b)} \\
SK'_{c,B} &= SK_{c,B}/H_2(ID_{AA_2}) \\
&= g_1^{s_2H_{x_2}(ID_c)}H_2(GID_B)^{H_{x_2}(ID_c)}
\end{aligned}
$$

where $GID_A = ID_{AA_1}||ID_A$ and $GID_B = ID_{AA_2}||ID_B$.

2) The ciphertext C is the same as in Step 5 of Section III-A except $x \in \{b, c\}$.
3) User A extracts $e(g_1, g_1)^{\lambda_b}$ as shown at the top of this page.
4) While user B extracts $e(g_1, g_1)^{\lambda_c}$ as shown at the top of this page.
5) Lastly, they compute M such that:

$$
\begin{aligned}
\frac{C_0}{\prod_x \left(e(g_1, g_1)^{\lambda_x}\right)^{c_x}} &= \frac{Me(g_1, g_1)^s}{e(g_1, g_1)^{\lambda_b c_b + \lambda_c c_c}} \\
&= \frac{Me(g_1, g_1)^s}{e(g_1, g_1)^s} \\
&= M
\end{aligned}
$$

where $\sum_x c_x A_x v = s$ for the constants $c_x \in \mathbb{Z}_N$.

IV. SECURITY FIXES

We discover that Yang *et al.*'s D-MA-ABE scheme shares a lot similarities to Lewko and Waters' Multi-Authority CP-ABE (MA-CP-ABE) scheme [2]. Comparing to Lewko and Waters' scheme, Yang *et al.*'s *AA* has an additional public key element PK_a; ciphertext has an additional element $C_{3,x}$; attribute secret key $SK_{i,u}$ has an additional element $H_2(ID_{AA})$ in it. It seems that these additional elements weaken the original scheme and lead to the two attacks above.

The key-only attack is caused by the lacking of authentication mechanism in the key issuing protocol. A quick

solution is to adopt a pairing-based proof of knowledge (PoK) protocol as the key issuing protocol instead of designing from scratch as done in [5]. PoK protocol provides authentication but not non-repudiation and it is suitable for the purpose of key issuing protocol here. The former allows the *AA*s to authenticate each other while the latter provides additional protection layer for user privacy as an *AA* can repudiate that itself as well as the underlying users are involved in the key issuing protocol. On the other hand, the collusion attack is resulted from the additional ciphertext component $C_{3,x}$. The solution is to remove this element and the $H_2(ID_{AA})$ element in $SK_{i,u}$ to adopt the secure decryption algorithm from Lewko and Waters' MA-CP-ABE.

Therefore, the complete solution is to extend Lewko and Waters' MA-CP-ABE scheme as suggested above to enable the key issuing mechanism, so that users can remain private to the *AA*s outside the domain as needed by a D-MA-ABE scheme. It remain as an open problem to tell whether a D-MA-ABE scheme can be trivially obtained from the combination of a MA-ABE scheme and a PoK protocol, or Yang *et al.*'s D-MA-ABE scheme is only a special case.

REFERENCES

[1] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1978.

[2] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology—Eurocrypt* (Lecture Notes in Computer Science), K. G. Paterson, Ed., vol. 6632. Berlin, Germany: Springer, 2011, pp. 568–588.

[3] Z. Liu, Z. Cao, and D. S. Wong, "Efficient generation of linear secret sharing scheme matrices from threshold access trees," IACR Cryptol. ePrint Arch., Bellevue, WA, USA, Tech. Rep. 374, 2010.

[4] Y. Yang, X. Chen, H. Chen, and X. Du, "Improving privacy and security in decentralizing multi-authority attribute-based encryption in cloud computing," *IEEE Access*, vol. 6, pp. 18009–18021, 2018, doi: 10.1109/ACCESS.2018.2820182.

[5] L. Zhang, P. Liang, and Y. Mu, "Improving privacy-preserving and security for decentralized key-policy attributed-based encryption," *IEEE Access*, vol. 6, pp. 12736–12745, 2018, doi: 10.1109/ACCESS.2018.2810810.